Інтернет програмування

лек 1

Основи JavaScript

Зав.каф.КН, PhD Граф М.С.

План курсу

Частина 1. Javascript
Частина 2. Vue.js

Іспит

Електронні ресурси для вивчення

https://w3schoolsua.github.io/js/
https://uk.javascript.info/

На іноземних мовах:
https://metanit.com/web/javascript/

Онлайн середовище розробки HTML, CSS, JS,
що працює в браузері:
https://jsfiddle.net/

Приклади роботи коду на JS

https://w3schoolsua.github.io/jstryit/tryjs_default.html
https://w3schoolsua.github.io/jstryit/tryjs_myfirst.html

Приклади взято з:
https://w3schoolsua.github.io/js/index.html#gsc.tab=0

JavaScript – Мова сценаріїв для надання інтерактивності веб-сторінкам.

Програми називаються ***скриптами***. У браузері вони підключаються безпосередньо до HTML і, як тільки завантажується сторінка - відразу виконуються.

Програми на JavaScript - звичайний текст.

Є як мінімум *три* чудові особливості JavaScript:
• · Повна інтеграція з HTML/CSS.
• · Прості речі робляться просто.
• · Підтримується всіма поширеними браузерами та включений за замовчуванням.

Версія	Офіційна назва	Опис
1	ECMAScript 1 (1997)	Перша редакція
2	ECMAScript 2 (1998)	Внесено редакційні виправлення.
3	ECMAScript 3 (1999)	Додані регулярні вирази. Доданий оператор try/catch.
4	ECMAScript 4	Ніколи не виходив.
5	ECMAScript 5 (2009)	Доданий "суворий режим". Додано підтримку JSON. Доданий String.trim(). Доданий Array.isArray(). Додано методи обходу елементів масиву.
5.1	ECMAScript 5.1 (2011)	Внесено редакційні виправлення.
6	ECMAScript 2015	Додані ключові слова let та const. Додані параметри за замовчуванням. Додано Array.find(). Доданий Array.findIndex().
7	ECMAScript 2016	Додано оператора зведення в ступінь (**). Доданий Array.prototype.includes.
8	ECMAScript 2017	Додано "паддинг" рядка (доповнення до потрібної довжини). Додано нові властивості об'єкта Object. Додано асинхронні функції. Додані пам'ять, що розділяється, і атомарні операції.
9	ECMAScript 2018	Додані властивості rest/spread. Додано асинхронні ітерації. Доданий Promise.finally(). Додавання до об'єкту RegExp.

Редакції ECMAScript

ECMAScript часто скорочується до ES.

Додаванн сценарію на сторінку:

```
<script type="text/javascript" > </script>
```

Приклад JavaScript-програми:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<p id='demo'>Клікніть кнопку, щоб змінити макет цього параграфа</p>

<button onclick='myFunction()'>Тицни мене!</button>

<script>
    function myFunction() {
        var x = document.getElementById('demo');
        x.style.fontSize = '25px';
        x.style.color = 'red';
    }
</script>
</body>
</html>
```

Прикріплення зовнішнього файлу:

```
<script type="text/javascript" src="example.js"></script>
```

Введення та виведення даних:

Метод – фрагмент коду багаторазового використання, призначений на вирішення загальних завдань.

alert

alert("Hello, World")

confirm

confirm(‘повідомлення’)

prompt

prompt("Введите Ваше имя, пожалуйста", "");

Однорядкові коментарі починаються з подвійного слеша //. Текст вважається коментарем до кінця рядка.

Багаторядкові коментарі починаються слішем-зірочкою "/*"і закінчуються зірочкою-слешем "*/".

Змінна

```
var message;
message =
'Привіт';
```

Константа - це постійна величина, яка ніколи не змінюється. Як правило, їх називають великими літерами через підкреслення. Наприклад:

```
var COLOR_RED = "#F00";
var COLOR_GREEN = "#0F0";
var COLOR_BLUE = "#00F";
var COLOR_ORANGE = "#FF7F00";
```

Імена змінних

На ім'я змінної JavaScript накладено кілька обмежень.

1.Ім'я може складатися з: літер, цифр, символів $ та _
2.Перший символ не може бути цифрою.
3.Як імена змінних не можна використовувати ключові слова та зарезервовані слова.

Ключові слова в JavaScript — це слова, які існують в ядрі JavaScript і вбудовані в його синтаксис, наприклад слово var.

Зарезервовані слова в JavaScript - це слова, які поки що не існують в ядрі мови JavaScript і не вбудовані в синтаксис, але в майбутньому, ці слова можуть бути впроваджені в ядро JavaScript, наприклад слово abstract.

Наприклад:

```
var myName;
var test123;
```

Типи даних

долар '$' і знак підкреслення '_' є такими самими звичайними символами, як літери.

Оператор typeof дозволяє дізнатися який тип даних присвоєний змінній, робиться це таким чином:

```
alert(typeof імяЗмінної);
```

Після цього скрипт повинен видати якесь
повідомлення: number, string, boolean, undefined, object.

1.Число number:

```
var n = 123;
n = 12.345;
```

Існують спеціальні числові значення **Infinity** (нескінченність) та **NaN** (помилка обчислень).

```
alert( 1 / 0 ); // Infinity
```

```
alert( "нечисло" * 2 ); // NaN, помилка
```

2. Рядок string:

```
var str = "Мама мила раму";
str = 'Одинарныі лапки також підійдуть';
```

3. Булевий (логічний) тип boolean.

```
var checked = true; // поле форми помічено прапорем
checked = false;    // поле формы не содержит прапор
```

4. Null - Спеціальне значення.

```
var age = null;
```

5. Undefined - Спеціальне значення, яке, як і null, утворює свій власний тип.

```
var u; alert(u); // виведе "undefined"
```

Можна привласнити undefinedі у явному вигляді, хоча це робиться рідко:

```
var x = 123;
x = undefined;
```

6. Об'єкти object.

Перші 5 типів називають *"примітивними"* .
Окремо стоїть шостий тип: *«об'єкти»* . Об'єкт (тобто член об'єктного типу даних) є колекцією значень (або елементарних, таких як числа і рядки, або складних, наприклад інших об'єктів). До нього відносяться, наприклад, дати, він використовується для колекцій даних та багато іншого.
Тип даних визначається після того, як змінній або константі було надано значення.

Операції в JS

4 основні види операцій **на JS** :
1. Операція присвоювання,
2. Арифметичні операції,
3. Операції порівняння,
4. Логічні операції.

Терміни: «унарний», «бінарний», «операнд»

Операнд - те, до чого застосовується оператор. Наприклад: 5*2 — оператор множення з лівим та правим операндами.

Унарним називається оператор, який застосовується до одного виразу.

оператор унарний мінус "-" змінює знак числа на протилежний:

```
var x = 3;
alert( -x );      // -3, унарний мінус
alert( -(x+2) ); // -5, унарній мінус застосовано до результату додаванн x+2
alert( -(-3) );  // 3
```

унарний "+" - нічого не робить в арифметичному плані, зате наводить операнд до числового типу

Бінарним називається оператор, який застосовується до двох операндів.

Бінарний мінус:

```
var x = 1, y = 3;
alert( y - x ); // 2
```

Бінарний плюс:

```
alert(2 + 2); // 4
```

Арифметичні оператори

Базові арифметичні оператори: плюс +, мінус –, помножити *, поділити /.

```
var i = 2;
i = (2 + i) * 3 / i;
alert(i); // 6
```

Арифметичний оператор %. Його результат **a % b** - це залишок від розподілу a на b.

```
alert(5 % 2); // 1, залишок від ділення 5 на 2
alert(8 % 3); // 2, залишок від ділення 8 на 3
alert(6 % 3); // 0, залишок від ділення 6 на 3
```

Додавання рядків, бінарний +

```
var a = "мій" + "рядок";
alert(a); // мій рядок
```

Якщо хоча б один аргумент є рядком, то другий також буде перетворений до рядка.

```
alert( '1' + 2 ); // "12"
alert( 2 + '1' ); // "21"
```

Інші арифметичні оператори працюють лише з числами і завжди наводять аргументи до числа.

```
alert( '1' - 2 ); // -1
alert( 6 / '2'); // 3
```

Унарний плюс +

широко застосовується, оскільки має корисний ефект - перетворення значення на число.

```
var a = "2";
var b = "3";
alert( a + b ); // "23", так к бінарний плюс додає рдки
alert( +a + b ); // "23", другий операнд - все ще рдок
alert( +a + +b); // 5, число, тому що обидва операнди попередньо перетворені до числа
```

Привласнення

```
var i = 1 + 2;
alert(i); // 3
```

Можливе привласнення по ланцюжку:

```
var a, b, c;
a = b = c = 2 + 2;
alert(a); // 4
alert(b); // 4
alert(c); // 4
```

Виклик x = вираз записує вираз x, а потім повертає його. Завдяки цьому присвоювання можна використовувати як частину складнішого виразу:

```
var a = 1;
var b = 2;
var c = 3 - (a = b + 1);
alert(a); // 3
alert(c); // 0
```

Пріоритет

Повна таблиця пріоритетів:
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

У виразі x = 4/2 + 2 є три оператори: присвоєння =, розподіл * і додавання +. Пріоритет множення дорівнює 5, воно виконається першим, потім відбудеться додавання +, у якого пріоритет 6, і після них - присвоєння =, з пріоритетом 17.

Інкремент/декремент: ++, -

```
var i = 2;
i++;      // більш короткий запис для i = i + 1.
alert(i); // 3
```

```
var i = 2;
i--;      // більш короткий запис для i = i - 1.
alert(i); // 1
```

Викликати ці оператори можна не лише після, а й перед змінною: i++ (називається «постфіксна форма») або ++i («префіксна форма»).

```
var i = 1;
var a = ++i; // (*)
alert(a); // 2
```

```
var i = 1;
var a = i++; // (*)
alert(a); // 1
```

Якщо результат оператора не використовується, а потрібно тільки збільшити/зменшити змінну – немає різниці, яку форму використовувати:

```
var i = 0;
i++;
++i;
alert(i); // 2
```

Якщо хочеться відразу використати результат, то потрібна префіксна форма:

```
var i = 0;
alert( ++i ); // 1
```

Якщо потрібно збільшити, але потрібне значення змінної до збільшення — постфіксна форма:

```
var i = 0;
alert( i++ ); // 0
```

Інкремент/декремент має вищий пріоритет і виконується раніше, ніж арифметичні операції:

```
var i = 1;
alert( 2 * ++i ); // 4
```

```
var i = 1;
alert( 2 * i++ ); // 2, буде виконано раніше, але значення повернеться старе
```

При цьому потрібно з обережністю використовувати такий запис, тому що при читанні коду часто не очевидно, що зміна збільшується. Три рядки — довші, наочніше:

```
var i = 1;
alert( 2 * i );
i++;
```

Подивіться, чи вам зрозуміло, чому код нижче працює саме так?

```
var a = 1, b = 1, c, d;
c = ++a; alert(c); // 2
d = b++; alert(d); // 1
c = (2+ ++a); alert(c); // 5
d = (2+ b++); alert(d); // 4
alert(a); // 3
alert(b); // 3
```

Оператори порівняння

Оператор	Операція	Типи значень
==	Перевірка рівності (еквівалентності)	будь-яке, будь-яке→булево
!=	Перевірка нерівності	будь-яке, будь-яке→булево
===	Перевірка суворої рівності (ідентичності)	будь-яке, будь-яке→булево
!==	Перевірка неідентичності	будь-яке, будь-яке→булево
<, <=, >, >=	Порівняння числових значень	число, число→булево
<, <=, >, >=	Порівняння рядків	рядок, рядок→булево

```
"a" > "Z"
// true
"9" < "a"
// true
```

```
"строка" > "стр0ка"
// true
```

Оператор ідентичності здійснює порівняння значень операндів без перетворення типів керуючись такими правилами:

- Якщо два значення мають різні типи, вони не є ідентичними.
- Якщо обидва значення null або undefined, вони ідентичні.
- Якщо обидва значення є true або false, вони однакові.
- Якщо одне або обидва значення NaN, вони не ідентичні.
- Якщо обидва значення є числами з тим самим значенням, вони ідентичні. 0 та -0 ідентичні.
- Якщо обидва значення є рядками і містять ті самі 16-бітові послідовності, вони ідентичні. Якщо рядки відрізняються довжиною або вмістом, вони не є ідентичними. Якщо два рядки виглядають однаково, але містять різні послідовності 16-бітових значень, вони не ідентичні.
- Якщо обидва значення посилаються на той самий об'єкт, масив або функцію, то вони ідентичні. Якщо вони посилаються на різні об'єкти (масиви чи функції), вони не ідентичні.

```
'7' === 7                  // false - різні типи даних
null === null              // true
undefined === undefined    // true
true === true              // true
false === false            // true
NaN === NaN                // false
(2 + 2) === 4              // true - два однакових числа
0 === -0                   // true
'строка' === 'Строка'      // false - перші символи рядка різні
```

Оператор еквівалентност. Якщо два значення мають однаковий тип, вони перевіряються на ідентичність.
Якщо значення операнда мають різні типи, оператор == виконує неявне перетворення типів і потім намагається виконати порівняння:

- Якщо одне значення null, а інше undefined, вони рівні.
- Якщо об'єкт порівнюється з числом або рядком, він перетворюється за допомогою методів valueOf або toString до примітивного значення - рядка або числа. Якщо конвертувати об'єкт не вдається - генерується помилка виконання.
- Якщо одне значення є числом, а інше - рядком, рядок перетворюється на число і виконується порівняння з перетвореним значенням.
- Якщо один з операндів має значення true, воно перетворюється на число 1, якщо значення false - на число 0.

```
null == undefined          // true
"123" == 123               // true
true == "1"                // true
false == 0                 // true
(4+2) == 6                 // true
"my car" == "my car"       // true
4 == 5                     // false
```

Оператори нерівності (!=) і неідентичності (!==) виконують перевірки, які протилежні операторам == і ===.
Наприклад, оператор нерівності != Повертає false, якщо два значення рівні один одному в тому сенсі, в якому вони вважаються рівними оператором ==, і true в іншому випадку.

```
4 != 3                    // true
"my car" != "My car"      // true
"4" != 4                  // false
"4" !== 4                 // true
NaN != NaN                // true
{x: 2} !== {x: 2}         // true
```

Виклик операторів із присвоєнням

```
var n = 2; n = n + 5; n = n*2.
```

Цей запис можна вкоротити за допомогою суміщених операторів:

Оператор	приклад	Еквівалент
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b
<<=	a <<= b	a = a << b
>>=	a >>= b	a = a >> b
>>>=	a >>>= b	a = a >>> b
&=	a &= b	a = a & b
|=	a |= b	a = a | b
^=	a ^= b	a = a^b

Ось так:

```
var n = 2;
n += 5; // топірю n=7 (працює як n = n + 5)
n *= 2; // топірю n=14 (працює як n = n * 2)
alert(n); // 14
```

Побітові оператори

Позначення	Операція	приклад
&	AND (Порозрядне І)	15 & 9 дає 9 (1111 & 1001 = 1001)
|	OR (Порозрядне АБО)	15 | 9 дає 15 (1111 | 1001 = 1111)
^	XOR (Порозрядне виключає АБО)	15 ^ 9 дає 6 (1111 ^ 1001 = 0110)
~	NOT (Заперечення)	~9 = -10 0000 0000 0000 0000 0000 0000 0000 1001 1111 1111 1111 1111 1111 1111 1111 0110
<<	LEFT SHIFT (Зсув ліворуч)	9 << 2 = 36 1001 << 100100
>>	RIGHT SHIFT (Зсув праворуч)	9 >> 2 = 2 1001 >> 10
>>>	ZERO-FILL RIGHT SHIFT (Зсув праворуч із заповненням нулями)	9 >>> 2 = 2 0000 0000 0000 0000 0000 0000 0000 1001 0000 0000 0000 0000 0000 0000 0000 0010 -9>>>2 = 1073741821 1111 1111 1111 1111 1111 1111 1111 0111 0011 1111 1111 1111 1111 1111 1111 1101

Логічні оператори

Оператор	Операція	Типи значень
&&	Логічне І	будь-яке, будь-яке ⟶ будь
||	Логічне АБО	-яке, будь-яке ⟶ будь -яке будь-
!	Логічне НЕ (інверсія)	яке ⟶ булево

```
var y = 1;
var x = 0 && ++y;  // ++y не буде обчислюватися
alert("x: " + x);
alert("y: " + y);
```

Якщо лівий операнд має значення, яке може бути перетворене на true, оператор && обчислює значення правого операнда і повертає значення правого операнда.

```
var x = 1 && 2;
var y = 1 && 0;
alert(x);  // 2
alert(y);  // 0
```

Оператор логічне АБО (||) виконує операцію АБО над двома операндами. Його робота починається з обчислення значення лівого операнда. Якщо лівий операнд має значення, яке може бути перетворено на true - повертається значення лівого операнда, при цьому значення правого операнда обчислюватися не буде:

```
var y = 1;
var x = 1 || ++y;  // ++y не буде обчислюватися
alert("x: " + x + "<br>");
alert("y: " + y);
```

Якщо лівий операнд має значення, яке може бути перетворене на true, оператор && обчислює значення правого операнда і повертає значення правого операнда.

```
var x = 0 || 2;
var y = null || 0;
alert(x + "<br>");  // 2
alert(y);           // 0
```

Події

Подія	Застосовується до	Виникає, коли	Обробник
Blur	вікна та всі елементи форми	користувач прибирає фокус уведення з вікна або елемента форми	onBlur
Click	кнопкам, radio-кнопкам, перемикачам/checkboxes, кнопкам submit та reset, гіперпосиланням	користувач клацає по елементу форми або кнопці	onClick
Focus	вікна та всі елементи форми	користувач передає фокус вікну або елементу форми	onFocus
Load	тілу документа	користувач завантажує сторінку в Navigator	onLoad
MouseOut	областям, гіперпосиланням	Користувач переміщує курсор за межі клієнтської карти зображень або гіперпосилання	onMouseOut
MouseOver	гіперпосиланням	користувач переміщає курсор над гіперпосиланням	onMouseOver
Resize	вікнам	користувач або скрипт змінює розмір вікна	onResize
Unload	тілу документа	користувач залишає сторінку	onUnload

Дякую за увагу!